Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

A Very Efficient and Accurate Pitch Detection Method

T. D. Teodorescu, D. F. Chiper[1]

Abstract – In the last 15 years, due to the expansion of the mobile telephony, but not only, efficient, low cost and low power compression methods for speech signals have been developed. At this moment, many standards for vocoders are implemented, the most used beeing GSM 723 or GSM 728. Most of them have as basic principle the linear prediction coding, which consists basically in coding a frame of typically 20ms of speech by using an MA predictor. This paper aims to improve the quality of a low bit rate vocoder (LPC10 like) by accurately finding the pitch.

Keywords: vocoder, LPC

I. INTRODUCTION

MA estimators of orders starting from 10 are sufficient to code the speech waveform for the large majority of speakers. At the receiver, it operates the inverse filter (AR) excited by a train of pulses for voiced frames or by white noise for unvoiced ones (LPC10 principle). Therefore, the only information that it is needed to be transmitted over the communication channel consists in 10 numbers (typically coded by means of LSP or LSF algorithm [4]), the gain of the filter and the period of the train of pulses (pitch period). At this moment, there are many standards that are using also the prediction error (for better quality of synthesized speech at the decoder), but with the cost of increasing the bit rate.

In order to obtain good quality for the synthesized speech at the receiver, it is a must to have a method which gives accurately the pitch for voiced frames. Although this subject is rather old, many researchers try to develop low cost methods for pitch computation. Starting from early 80's, researchers such as Rabiner, Marchandt or Spanias [1-2, 5] have proposed either time-domain methods or frequency domain approaches for pitch estimation.

Our method uses time-domain approach, and correlation method, by using AMDF measure for waveform similarity. This method implies only subtractions and additions and do not make use of multiply-accumulator blocks taking advantage of very simple chip architectures (fixed point), dedicated or not. In this paper we propose a few methods for improving AMDF performance by using signal pre-filtering and a pitch control mechanism that significantly improves the AMDF performance, at very low computation cost.

It is well-known that the regions of speech for which it is hard to detect the pitch are transition regions between voiced and unvoiced frames and also voiced frames of low amplitude. These are the cases where the signal is far from being quasi-periodic; our method improves the performance of the vocoder in these cases.

In the last part of the paper we will give an example of pitch detection engine working within a quasi-standard LPC codec and also an example by using a new vocoder principle, with good performance strongly related to accurate pitch detection. The last vocoder principle has the advantage of lower sensitivity to noise compared with standard codecs.

II. LPC BASED CODER PRINCIPLE

The architecture of the coder is given below:

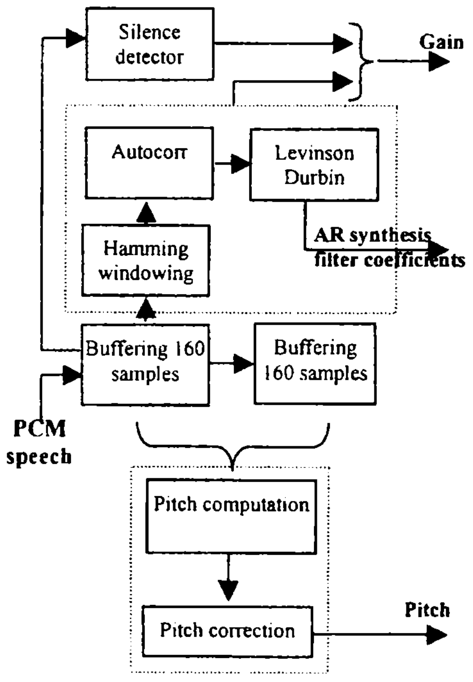


Fig. 1. LPC coder

The coder contains two buffering levels of 160 samples each. First previous buffer contains only

[1] Faculty of Electronics and Telecommunications, Bd. Carol, no. 11, Iasi, e-mail: t-teodor@etc.tuiasi.ro

zeroes. Then the current buffer is filled with the content of the previous buffer and so on. The speech model for a frame (AR synthesis filter coefficients and the gain) is built by using the samples from the previous buffer, while for the pitch calculation we use two buffers: previous and current.
Since the model computation block is standard, we will focus on the pitch block. If this block returns zero, then the previous buffer (for which the whole parameters are determined) is declared as being unvoiced. Otherwise, it is a voiced buffer and the synthesis will be done accordingly.
In the following figure the pitch computation block is detailed:

The "period" of the analyzed signal corresponds in this case to the minimum of the curve (excepting the minimum in zero). The AMDF is an excellent substitute for the usual autocorrelation function, since it makes use only of the subtractions, additions and modulus and practically the algorithm complexity tends to zero.
The averaging window block is meant to eliminate the false local minima, improving this way the accuracy of local minima detection.
If, in the interest interval for the pitch[2] (20..160), the minimum is sufficiently big comparing to the maximum, the buffer is voiced. Otherwise, it is unvoiced (pitch=0).

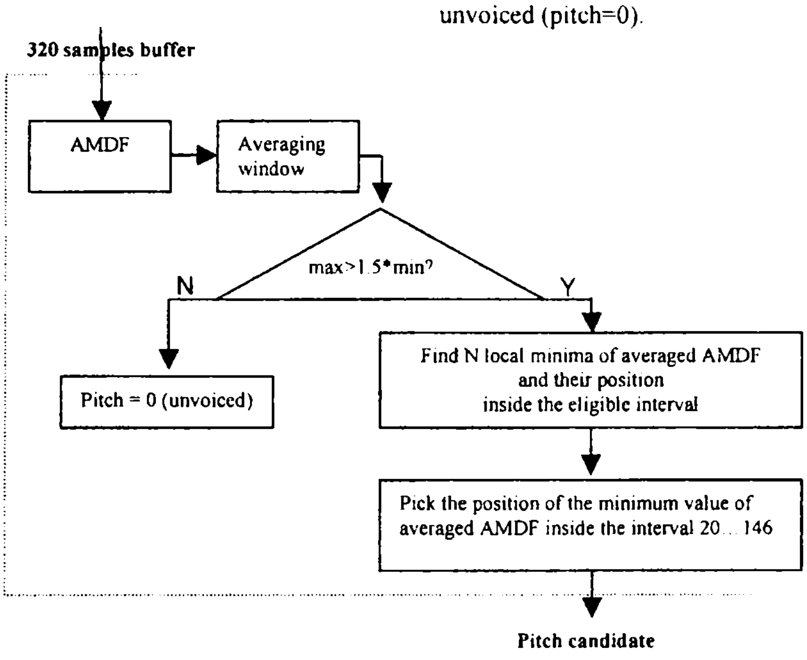

320 samples buffer
AMDF
Averaging window
max>1.5*min?
N
Y
Pitch = 0 (unvoiced)
Find N local minima of averaged AMDF and their position inside the eligible interval
Pick the position of the minimum value of averaged AMDF inside the interval 20...146
Pitch candidate

Fig. 2. Pitch computation

The AMDF (Average Mean Distance Function) is mainly used to calculate the fundamental frequency of a signal. The starting point of the segment is chosen and compared with the value of the signal in a certain distance and the difference between the absolute values is calculated. This is done for all points and the differences are accumulated. This is repeated for several distances. The x-axis reflects these distances.
One typical AMDF is given below:

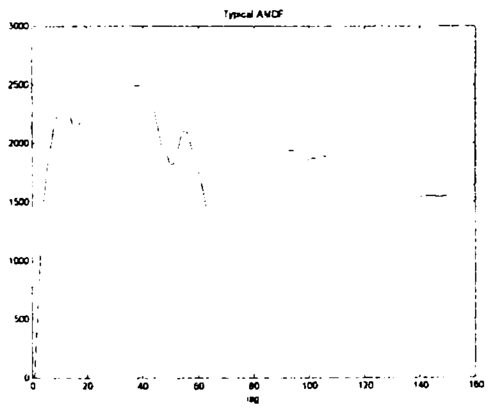


Fig. 3. Typical AMDF with pitch 65

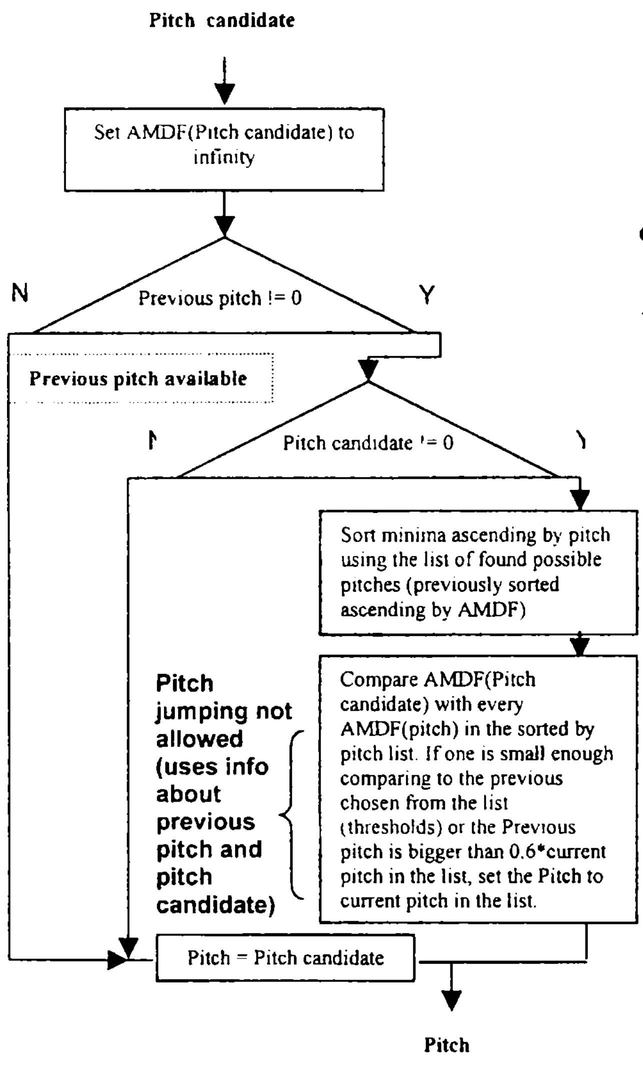
In the case the buffer is voiced, a preset number of local minima (maximum N is around 50 for a buffer of 160 samples) is searched inside the interval 20...160. A list with found minima is built. The position that corresponds to the global minimum is the most probable pitch value and this is the simple case, when there are no closed pitches as AMDF values. We named this value the pitch candidate.
The block that follows (see Fig. 1) is the Pitch correction block. It handles the situation when the pitch candidate is not the real pitch. It is a common sense fact that during the natural speech the voice of a person doesn't have pitch glitches or jumps. Starting from this statement, the block that follows tries to bring the real pitch into the scene, even though, maybe, it doesn't correspond to the minimum AMDF value, but to some value closed to that.
The Pitch correction block makes use of the Previous pitch, which is the pitch of the previous buffer, if that was a voiced one. If not, "the mechanism" has no data for further assumptions and the pitch is in fact the pitch candidate. Therefore, the Pitch correction block

[2] Time corresponding to the "fundamental frequency" for the speech buffer

has no object of improvement. Also, if the pitch candidate is zero (this means that the current buffer is unvoiced) the situation is the same.

If neither Previous, nor Candidate pitch is 0, than we are situating in the case when have a sequence of voiced buffers (obviously belonging to the same person) and we should control the pitch in order not to have glitches or pitch jumps.

In that case, we are building another list of pitches and their corresponding AMDF values (based on the sorted by AMDF value list found in the previous block , but this time sorted by itch value. The AMDF of the pitch candidate was previously set to "infinity", in order have zero influence on this part of the algorithm. The AMDF(Pitch candidate) and the Pitch candidate are stored in other location than in the list.



Pitch candidate
Set AMDF(Pitch candidate) to infinity
N
Previous pitch != 0
Y
Previous pitch available
N
Pitch candidate != 0
Y
Sort minima ascending by pitch using the list of found possible pitches (previously sorted ascending by AMDF)
Compare AMDF(Pitch candidate) with every AMDF(pitch) in the sorted by pitch list. If one is small enough comparing to the previous chosen from the list (thresholds) or the Previous pitch is bigger than 0.6*current pitch in the list, set the Pitch to current pitch in the list.
Pitch jumping not allowed (uses info about previous pitch and pitch candidate)
Pitch = Pitch candidate
Pitch

Fig. 4 Pitch correction

If we encounter a situation when the AMDF(Current pitch in the list) is smaller than the AMDF(Pitch candidate), we see then if the AMDF(Current pitch in the list) is close enough with AMDF(Pitch candidate) or if the Previous pitch is bigger than 0.6*Current pitch in the list (this will not allow pitch jumps with more than 67%). If so, we change the Pitch candidate and AMDF(Pitch candidate) to the Current pitch in the list and its AMDF value and the algorithm continues until the all the elements in the list sorted ascending by pitch are used. The Pitch is the last Pitch candidate.

III. LPC BASED DECODER

The decoder architecture is given below:

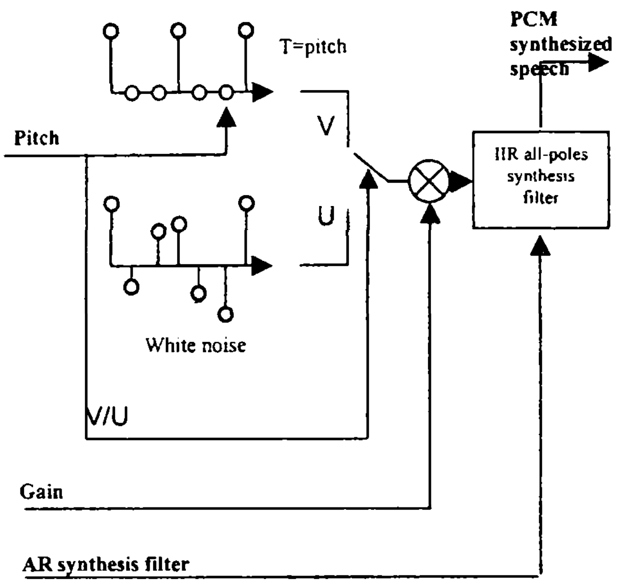


Fig. 5. LPC decoder

At the decoder part (receiver) the Pitch, the Gain and the AR synthesis filter coefficients are received as bit stream packed in a certain way. They are parsed and the values are re-calculated at the receiver.

If the Pitch is zero, the buffer will be synthesized as the response of the IIR filter to white noise (flat spectrum). The spectrum of the unvoiced will be then the result of the spectral shaping of the filter over the spectrum of the white noise.

If the Pitch is not zero, then the stimulus for the filter will be a train of impulses with the period of the pitch. If the previous buffer was also voiced, we take care not to start the pulses from the beginning of the buffer in order to obtain a smooth transition between two buffers of the same or other pitch. Also the gain is increased/decreased in a linear manner, also for smooth transitions between buffers.

The zero value for the gain corresponds to silence detected on the coder side.

IV. EXPERIMENTAL PROOF

In the following section we will give an example of concept proof on a standard PCM file.

The sample rate is 8 KHz, with a depth of 8 bits/sample. The vocoder is built using a 10 order AR synthesizer, working on buffers of 160 samples (a buffer is 20 ms long). There is no quality difference between the non-quantized version and the 3800 bps (LSF) version of this non-standard vocoder.

We have proved this way that the quality of low-bit rate vocoders is given by the accuracy of the pitch and the smoothness of the transitions between unvoiced-voiced-unvoiced sections and up-to a certain bit rate, the quality does not depend on the number of the bits the information sent on the communication channel.
Below there is the pitch representation over time without the pitch correction mechanism (Fig. 6a) and with the correction mechanism (Fig. 6b):

From these results it is clear that the pitch curve is smoother in the pitch control version of the algorithm than in the one without pitch control.
The Pitch control algorithm is not an intrusive one (it is well-known that when taking account the history, if not carefully, ones can bias the voice synthesis).
For example, the spike around buffer 200 is eliminated and many more.

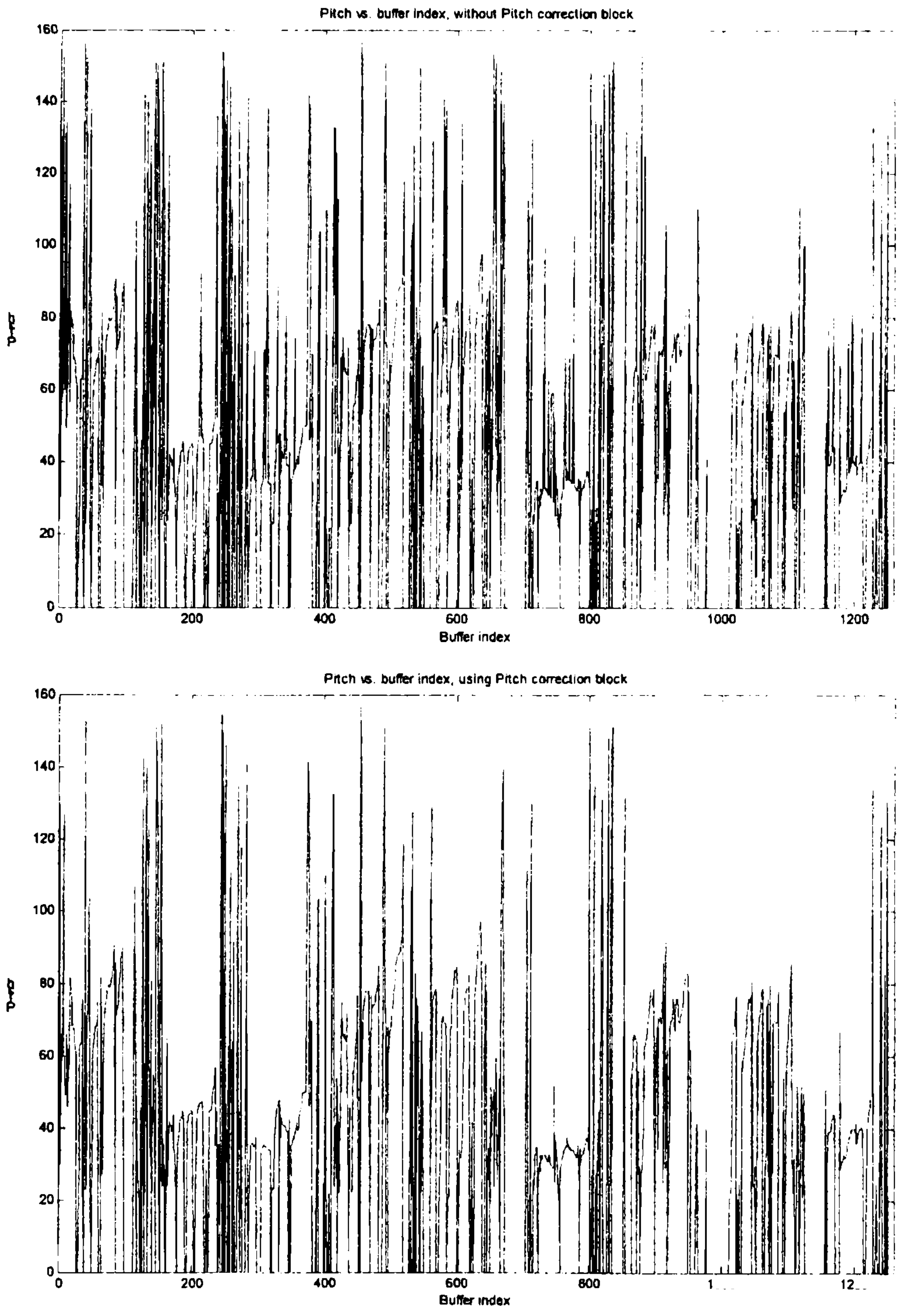


Fig. 6 a) Without pitch correction, b) With pitch correction

In the following we will present the AMDF function for buffers 212, 213, 214 and 215, corresponding to the zone where the pitch control algorithm corrects the two errors:

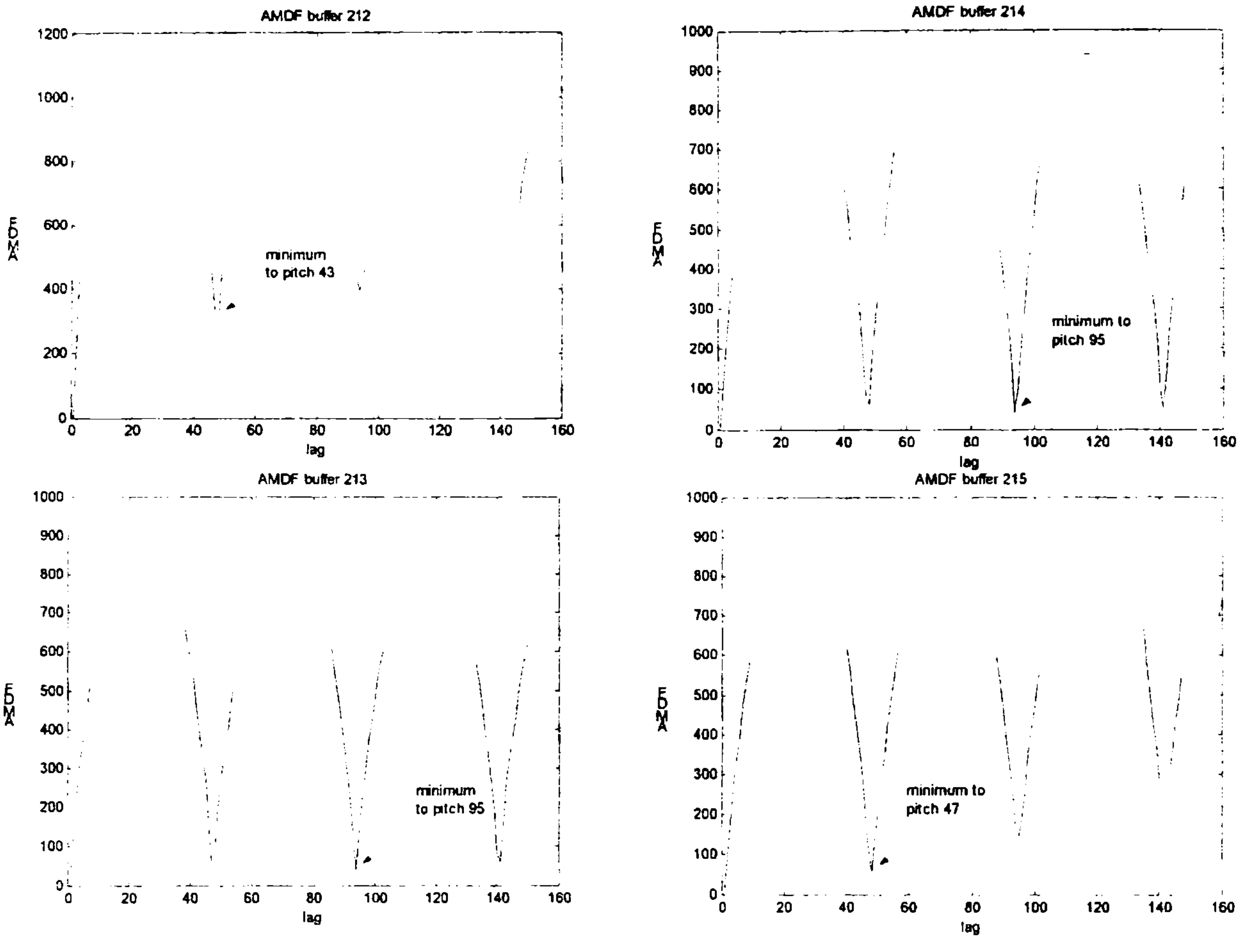


Fig. 7 AMDF for 4 consecutive buffers

It can be easily seen that the most probable mistake that can be made is to pick the double of the pitch and not the true pitch. Therefore, the algorithm was designed in this manner: to not allow the pitch jumps when the AMDF value is similar to two or more values.

Due to the algorithm proposed by us, the pitch doesn't jump (see . ig. 6b) nei.he. in .his si...a.ion, no. in many others.

In Fig. 8a) there is presented the result after speech synthesis in the case of the algorithm working without Pitch correction block and in Fig. 8b) there is the result when the pitch is corrected based on the pitch history. The fact that impulses have double period in the first case has a great negative impact on the quality of synthesized speech.

V. FURTHER WORK

Starting from the idea of having accurate pitch dete..ion, a MIDI-lik. voco..r can b. built. W. can simply determine the pitch for a buffer, sample the original waveform between two consecutive "pulses" and send to the receiver that minimum information. The vocoder have a greater sensitivity to pitch errors than the vocoder described here, but it is an idea to be taken into account when developing low bit rate vocoders.

VI. ACKNOWLEDGEMENT

The authors would like to thank Mr. Virgil Mager from IT Group Romania for the ideas exchange during this project.

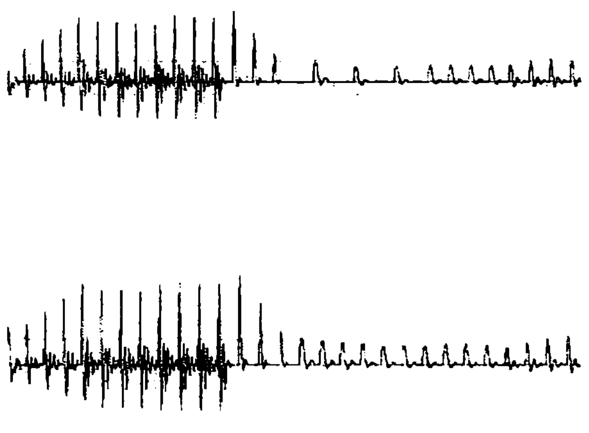


Fig. 8 a) Synthesized speech with wrong pitch
b) Correctly synthesized speech

REFERENCES

[1] M. R. Sambur, A. E. Rosenberg, L. R. Rabiner and C. A. McGonegal, "On Reducing the Buzz in LPC Synthesis, , Conference Record 1977" *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Hartford, Connecticut, pp. 401-404, May 1977.

[2] C. A. McGonegal, L. R. Rabiner and A. E. Rosenberg, "A Subjective Evaluation of Pitch Detection Methods Using LPC Synthesized Speech", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-25, No. 3, pp. 221-229, June 1977

[3] S.K. Mitra, *Digital Signal Processing - A Computer Based Approach*, Second Edition, McGraw Hill Inc., 2002

[4] P. Kabal, R. P. Ramachandran, "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials", *IEEE Trans. On Acoustics, Speech and Signal Processing*, vol. ASSP-34, No. 6, Dec 1986, pp. 1419-1426.

[5] A. S. Spanias, "Speech Coding: A Tutorial Review", *Proc. Of IEEE*, vol. 82, no. 10, October 1994, pp. 1541-1581.

[6] http://www.hawksoft.com/hawkvoice/codecs.shtml.